UNIT 1

INTRODUCTION TO THE ALGOTITHMICS

Patricia Cuesta Ruiz

Isabel Blanco Martínez

EXERCISE 3

Analyze the efficiency of the following code:

```
fun Calculo(x,y,z: entero) dev valor:entero
var i,j,t: entero
  valor ← 0
  Desde i ← x hasta y Hacer valor ← valor + i fdesde
  si (valor ÷ (x+y)) <= 1 entonces Devolver z
  si no
      t ← x + ((y-x) ÷ 2)   { ÷ es la división entera }
      Desde i ← x hasta y Hacer
            Desde j ← (3*x) hasta (3*y) Hacer
                  valor ← valor + Minimo(i,j)
            fdesde
      fdesde
      valor ← valor + 4*Calculo(t,y,valor)
      Devolver valor
  fsi
ffun
```

$$n = y - x$$

We're using *y – x* as the value of n because the amount of loop iterations is *y - x*, except the inner one, which is *3n*.

$$T(n) = 1 + \sum_{i=1}^{n} \sum_{i=1}^{3n} 1 + T\left(\frac{n}{2}\right) + 1$$

$$y - t = y - x - \left(\frac{y-x}{2}\right) = \frac{-y + x + 2y - 2x}{2} = \frac{y-x}{2} = \frac{n}{2}$$

After seeing that *y - t* is equal to $\frac{n}{2}$, the recursive call can be replaced by T($\frac{n}{2}$).

$$n = 2^k \qquad T(n) = x^k$$

$$T(n) = 3n^2 + n + 4 + T(\frac{n}{2})$$

$$x^k = 3 * 2^{2k} + 2^k + 4 + x^{k-1}$$

$$x^k - x^{k-1} = 3 * 2^{2k} + 2^k + 4$$

Homogeneous equation:

$$x^{k-1}(x-1) = 0 \qquad x^H = A * 1^k$$

roots: *x = 1*

Particular equation:

$$x^{k-1}(x-1) = 3 * 2^{2k} + 2^k + 4 \qquad x^P = B * 4^k + C * 2^k + D * k * 1^k$$

roots: *x = 4*

x = 2

x = 1

Finally:

$$x = A + 4^k B + 2^k C + kD \qquad n = 2^k \rightarrow k = \log_2(n)$$

$$T(n) = x = A + Bn^2 + Cn + D\log_2(n)$$

Complexity of **O(n^2)**

EXERCISE 5

Program a function to determine if a number received as parameter is prime. Analyze the efficiency and complexity.

```
def isPrime(num):
    b = True
    for i in range(2, num):
        if num % i == 0:
            b = False
    return b

print(isPrime(12))
```

$$T(n) = 1 + \sum_{i=1}^{n} 1 + \max\{1,0\} = 1 + \sum_{i=1}^{n} 1 + 1 = 1 + 2n$$

Complexity of **O(n)**

EXERCISE 8

Program a recursive procedure to obtain the inverse number of a given one.

Example: 627 → 726

Analyze the efficiency and complexity.

```python
def inverseNumber(n):
    numberInv = ''
    if (int(n/10) == 0):
        numero = str(n%10);
        numberInv += numero
    else:
        numero = str(n%10);
        numberInv += numero + inverseNumber(int(n/10))
    return numberInv

print(inverseNumber(678))
```

After seeing that n/10 is equal to n-1, the recursive call can be replaced by T(n-1).

$$T(n) = 1 + 1 + \max\{1, 1 + T(n-1)\} = 2 + 1 + T(n-1)$$
$$T(n) = 3 + T(n-1)$$
$$T(n) = x^n$$

$$x^n = 3 + x^{n-1}$$
$$x^n - x^{n-1} = 3$$
$$x^{n-1}(x-1) = 3$$

Homogeneous equation:

$$x^{k-1}(x-1) = 0 \qquad x^H = A * 1^k$$

roots: $x = 1$

Particular equation:

$$x^{k-1}(x-1) = 3 \qquad x^p = B * n * 1^k$$

roots: $x = 1$

We multiply by n because the value in both roots is 1

Finally:

$$T(n) = x = A + Bn$$

Complexity of **O(n)**

EXERCISE 6: Extra Exercise

Program a function to determine if a number received as parameter is perfect. Analyze the efficiency and complexity.

```
def isPerfect(num):
    n = 0
    for i in range(1, num):
        if num % i == 0:
            n = n + i
    return n == num

print(isPerfect(6))
```

$$T(n) = 1 + \sum_{i=1}^{n} 1 + \max\{1,0\} = 1 + \sum_{i=1}^{n} 1 + 1 = 1 + 2n$$

Complexity of **O(n)**

EXERCISE 7: Extra Exercise

Write a program which ask a positive number to the user (N) and obtain how many prime numbers there are between 1 and that number N, and how many perfects between 1 and N. Analyze the efficiency and complexity.

```python
def isPrime(num):
    b = True
    for i in range(2, num):
        if num % i == 0:
            b = False
    return b

def isPerfect(num):
    n = 0
    for i in range(1, num):
        if num % i == 0:
            n = n + i
    return n == num


n = int(input("Write down a number: "))
nPrimes = 0
nPerfects = 0
for i in range (1, n):
        if (isPrime(i) == True):
            nPrimes = nPrimes + 1
        if (isPerfect(i) == True):
            nPerfects = nPerfects + 1
print("Between 1 and", n, ":\nNumber of Perfects: ", nPerfects, "\nNumber of Primes: ", nPrimes)

```

$$T(n) = 1 + 1 + 1 + \sum_{i=1}^{n}\Big(1 + \underbrace{\Big(\sum_{i=1}^{n} 1 + 1\Big)}_{\text{isPrime(i)}} + 1 + 1 + \underbrace{\Big(\sum_{i=1}^{n} 1 + 1\Big)}_{\text{isPerfect(i)}} + 1\Big) + 1$$

$$T(n) = 4 + \sum_{i=1}^{n}(4 + 2n + 2n)$$

$$T(n) = 4 + \sum_{i=1}^{n}(4 + 4n)$$

$$T(n) = \ 4n + 4n^2 + 4$$

Complexity of **O(n^2)**